Visual Modelling with Rational Rose and UML

Sridhar Pandurangiah
Director – Engineering
sridhar@sastratechnologies.in

Sastra Technologies

What this training is not!

- Not a tutorial on analysis and design
- Not an explanation on notation and semantics of UML
- Not a training on any particular architecture for a particular language e.g. Java

What is this training about?

- Introduction to concepts needed to visualise a software system – process, notation, tool
- Guidance on Visual Modelling (Rational Objectory Process), Notation (UML), tool (Rational Rose)
- How to use the three in tandem in an iterative and incremental manner

INTRODUCTION

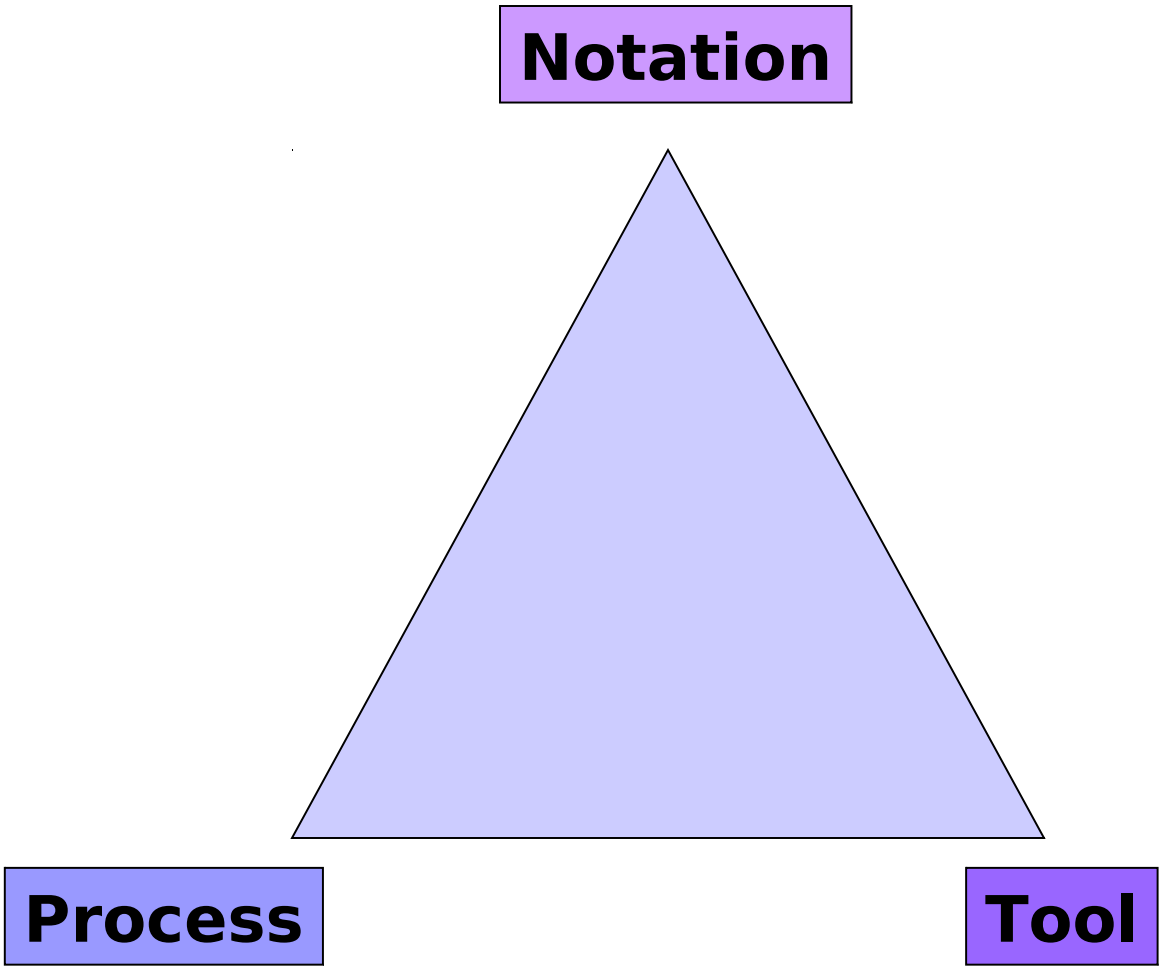
What is visual modelling?

- Way of thinking about problems using models organised around real world ideas
- Models are abstractions that portray the essentials of a complex problem by filtering out the non essentials
- Abstraction is a fundamental human capability that permits us to deal with complexity

What is visual modelling?

- We build models because we cannot comprehend such systems in their entirety (E.g. Shed Vs House Vs Skyscraper)
- Models help us organise, visualise, understand and create complex things.

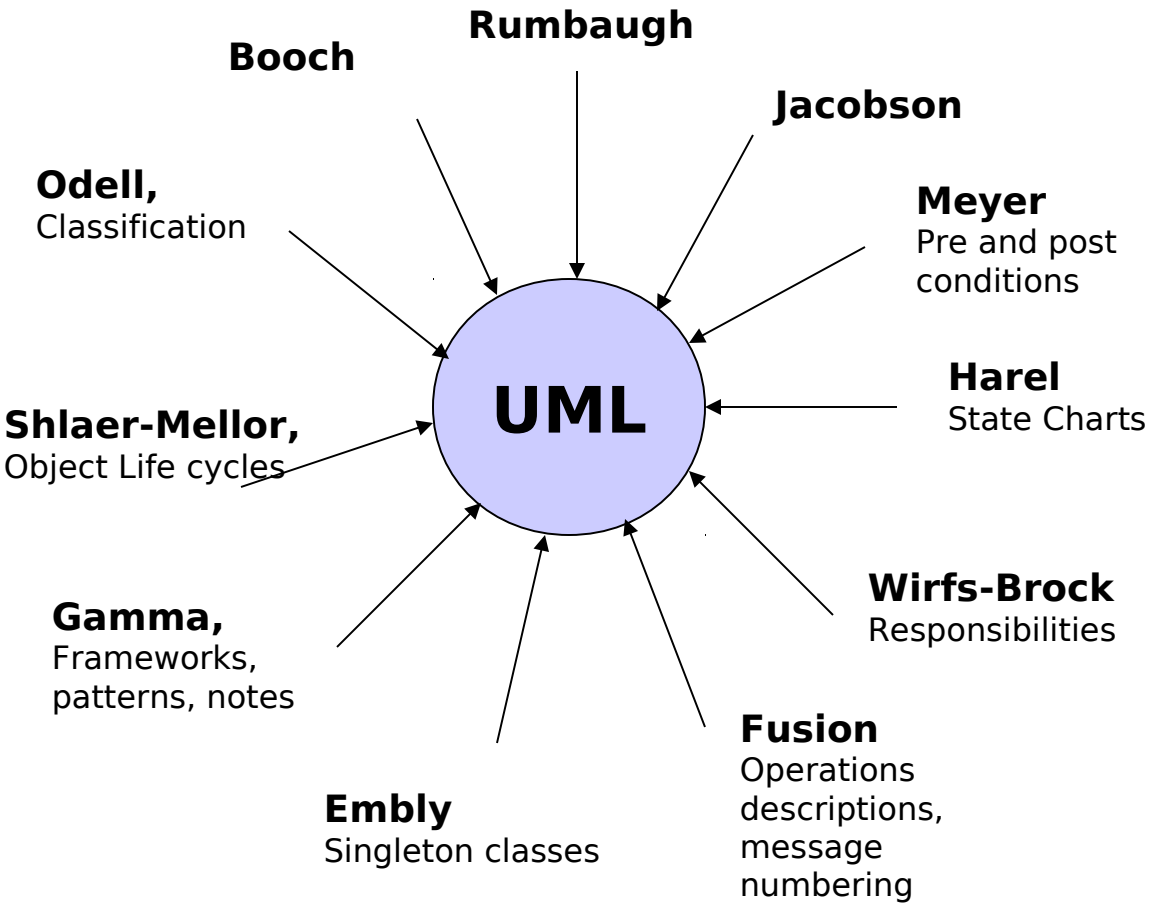
Triangle for success

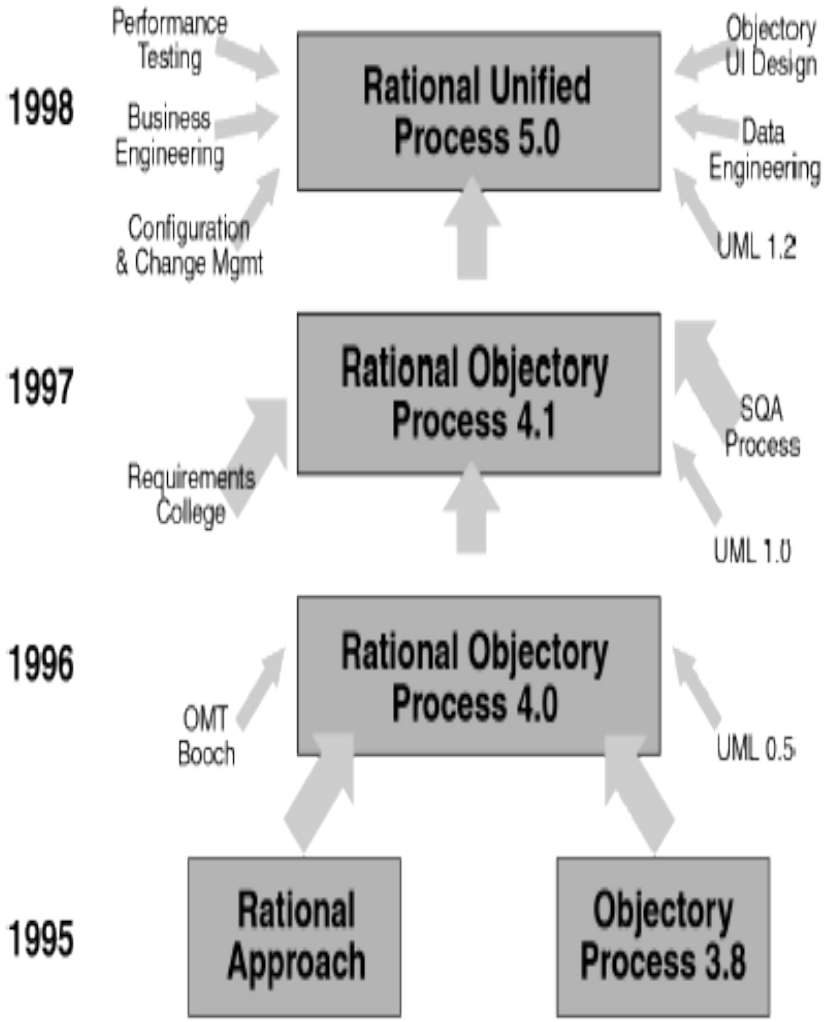


Role of Notation

- Serves as a language for communicating decisions
- Provides semantics to capture strategic and tactical decisions
- Offers a form concrete enough for humans to comprehend and for tools to manipulate

Brief history of UML



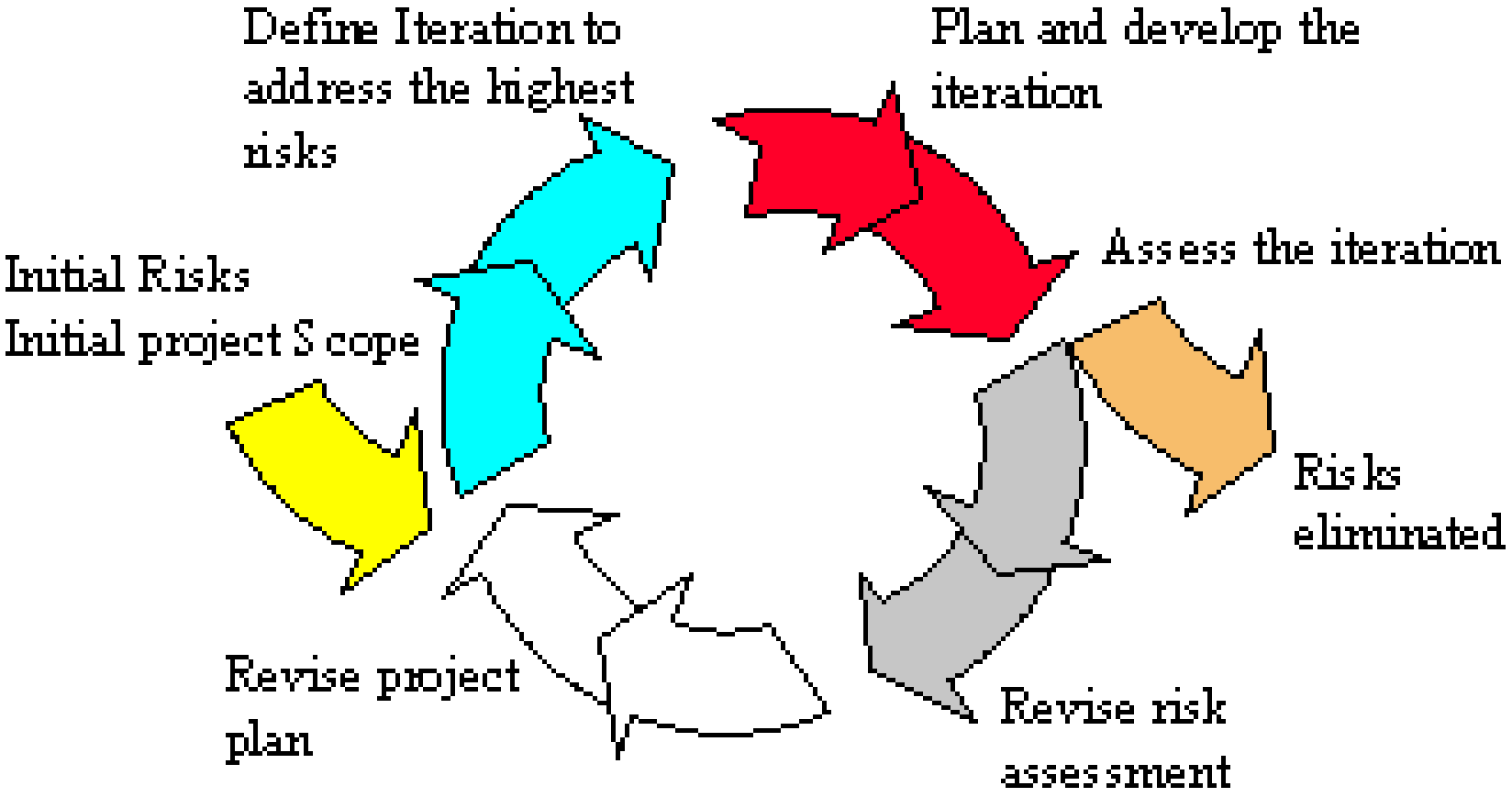


Genealogy of the Rational Unified Process

Role of Process

- Successful development project satisfies or exceeds customers expectations, developed in timely or economical fashion and is resilient to change and adaptation
- Development life cycle must promote creativity and innovation
- Development process must be controlled and measured to reach closure

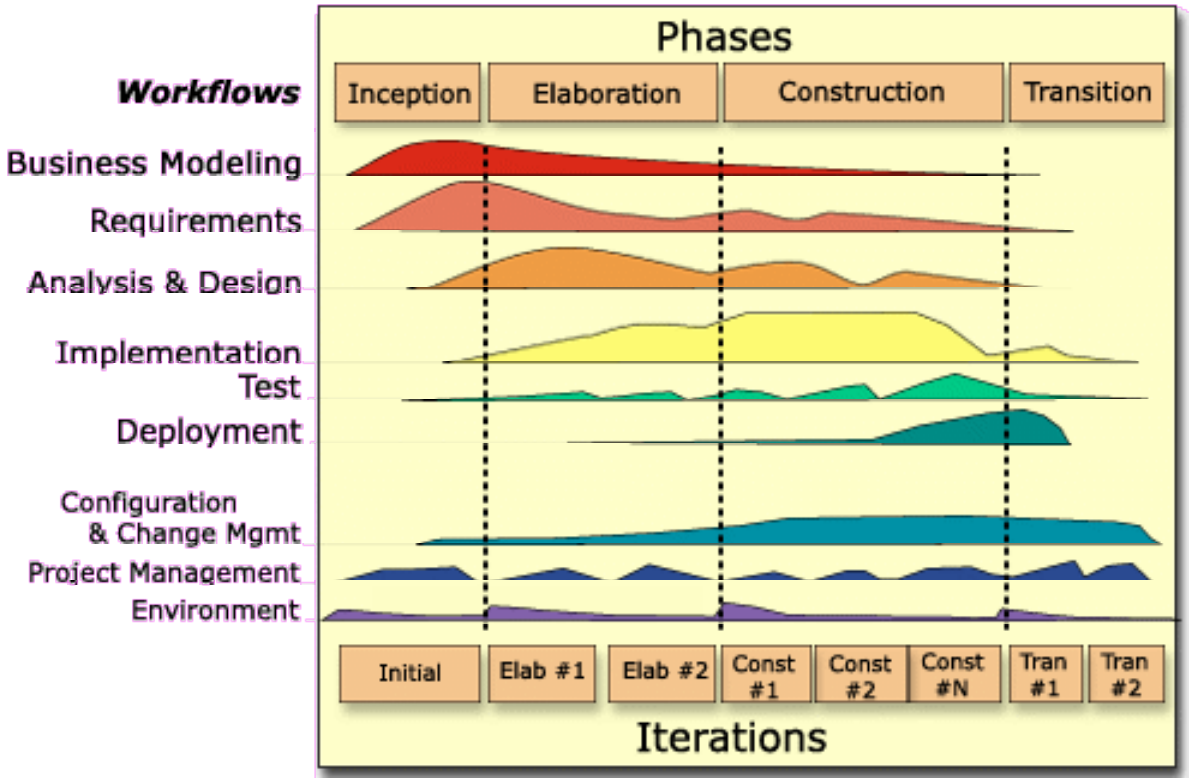
Iterative and Incremental Development



The Rational Objectory Process

- Provides control for iterative and incremental lifecycles
- Structured along two dimensions
  - Time : lifecycle division into phases and iterations
  - Process Components : artifacts and well defined activities

Process Components applied to each time based phase



Rational Rose® - Tool Mentor

- Identify and design business objects and map them to software components
- Partition services across a three tiered service model
- Design how components will be distributed across a network
- Generate code from the model

Rational Rose® - Tool Mentor

- Reverse engineer components / code to models
- Roundtrip engineering to synchronise designs with code

BEGINNING A PROJECT

Inception Phase

- “The System we want does ……”
- Vision for the idea : customers, domain experts, other developers, industry experts, feasibility studies, review of existing systems, brainstorming, research, trade-studies, cost-benefit analysis
- Validate assumptions
- Identify risks
- Identify external interfaces
- Consider business needs, available resources,
- First cut resource and schedule planning

Business Credit Disbursal

- The Imperial Bank of India has introduced a line of credit for small and medium business called”BCD : Business Credit Disbursal”. Your assignment is to model the system

CREATING USE CASES

System Behaviour

- Functionality documented in use cases model
- Use cases – functions
- Actors – surroundings
- Relationship between use cases and actors – use case diagrams

Actors

Actor

- ACTORS are not part of the system – they interact with the system
- Identify actors – man/machine who provide or receive information
- Use role, designation, nouns to identify actors
- Arrive at the final list iteratively

Exericse

- Identify the *Actors* in Business Credit Disbursal

Actors in Business Credit Disbursal

- Telemarketing executives
- Direct Marketing Executives
- CreditOfficer
- Prospect

Use Cases

UseCase

(from Use Case View)

- Model a dialogue between an actor and the system
- Sequence of transactions that yield a measurable result of values to an actor
- Use verbs and adverbs to identify an usecase

Exercise

- Identify the *use cases* in Business Credit Disbursal

Use cases in Business Credit Disbursal

- Telemarketing executive
  - Fix appointment with customer
  - Handle customer queries
- Direct Marketing Executive
  - Meet Customer and explain credit products
  - Collect application mandatory documents
  - File application with credit officer
  - Followup with customer
- Customer
  - Get application forms filled in
  - Enquire about various schemes
  - Enquire about status of application

Use case : Flow of events

- Is the events needed to accomplish the required behavior
- Should be written in terms of what the system should do and not how the system does it
- Written in the language of the domain, not in terms of implementation

Use case relationships…

- Association relationship may exist between an actor and a use case – *communicates association*
- Association may be navigable in one or both directions.
- Arrowhead on the association line depicts direction
- Association is the only type of relationship allowed between an actor and a use case

Use case relationships

- *Uses relationship*
  - Same functionality that's used in many use cases can be moved to a separate use case e.g. user verification, accounting entries generation, overrides, limits & exposure tracking etc
- *Extends relationship*
  - Used to depict optional behavior
  - Behavior triggered under exceptional circumstances
  - Several different flows that may run based on actor selection

Stereotypes

- Used to extend basic modelling elements to create new elements
- Depicted using gullemets (<< >>)
- Placed along the relationship line

Notation of use case relationship

Use case diagram

- Graphical view of some or all of the actors, use cases and their interactions for a system
- Main use case diagram – Depicts system boundary
- Other use case diagrams may be created
  - A diagram showing all use cases for an actor
  - A diagram showing all uses cases being implemented in an iteration
  - A diagram showing single use case and all its relationships

FINDING CLASSES

Object ?

- Representation of an entity – real world or conceptual e.g. computer, credit history
- Concept, abstraction, or thing with well-defined boundaries and meaning
- Characteristics : state, behavior and identity

State, behavior, identity?

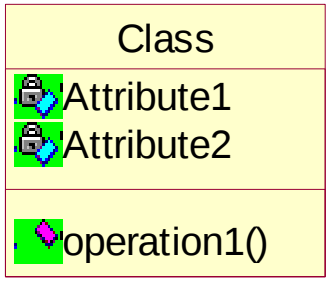
OBJECT NOTATION TO COME HERE

- State : One of the possible conditions in which an object exists
- Object is defined by its attributes
- Behavior is how an object responds to requests to other objects and typifies everything an object can do
- Identity means that each object is unique

Class ?

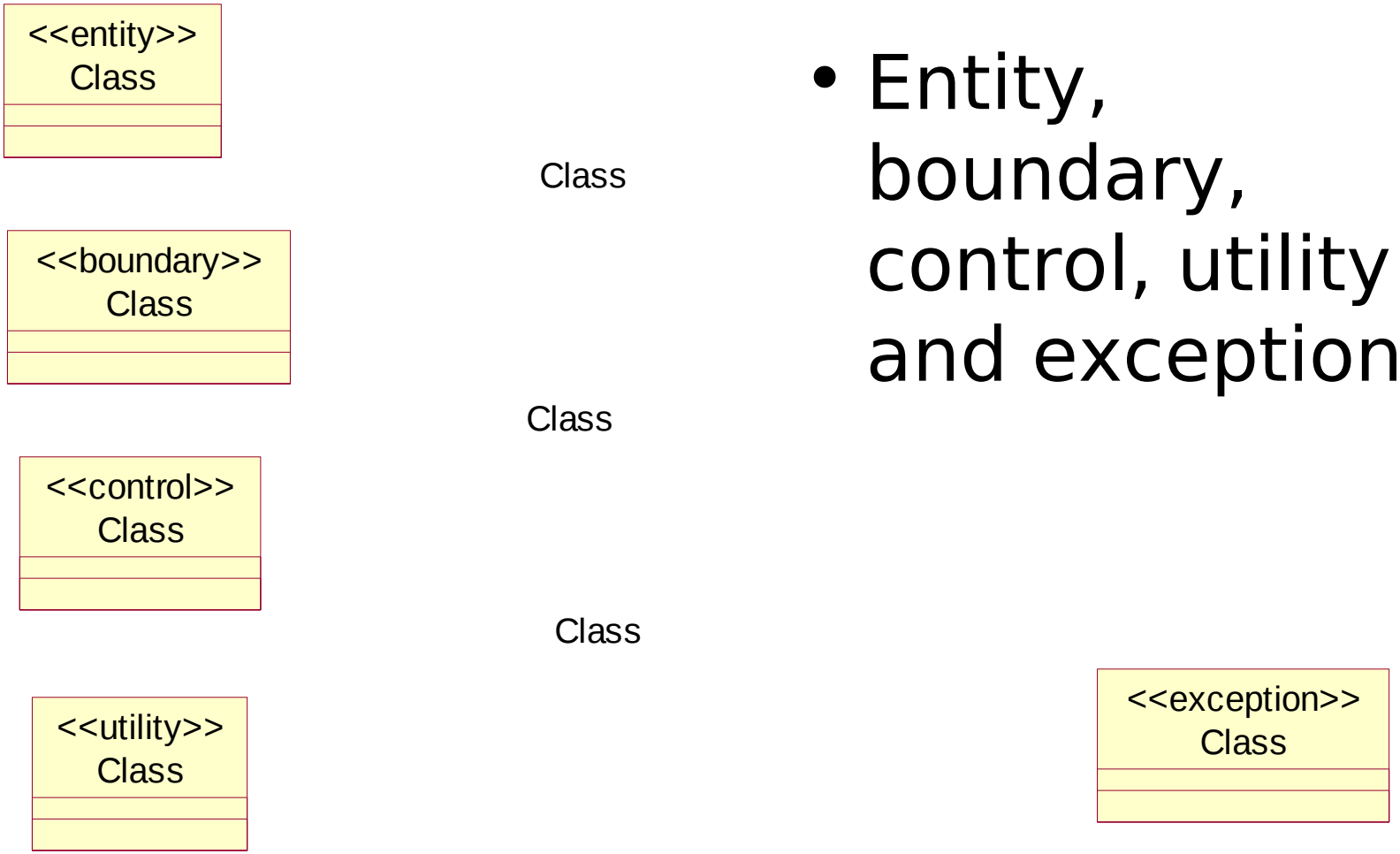
- Group of objects with common properties (attributes), common behavior (operations), common relationships to other objects and common semantics
- A good class captures one and only one abstraction
- Name classes using vocabulary of the domain using a singular noun that best characterises the abstraction

Class ?



- Using an acronym only if it means the same to all (but full name should be contained in the class) – PAN vs OPORSONDIP!

Stereotypes and classes



- Entity, boundary, control, utility and exception

Class Notation

Interface Class

Table Class

Domain Class

Table Class

Discovering Classes

- There is no cookbook
- “This is hard” – Grady Booch
- Follow an iterative process
- Initial set of classes may not be the one that gets implemented
- So the term *Candidate Class* – First set of classes found for a system

Entity Class…

- Models information and associated behavior that is generally long lived
- May reflect a real world entity or perform tasks internal to the system
- Independent of their surroundings I.e. not dependent on how surroundings communicate with the system.
- Many times they are application independent I.e. they can be reused in other systems

Entity class

- Filter list of nouns in the use case document
- Remove nouns that are outside the problem domain, nouns that are language expressions, nouns that are descriptions of class structures
- Called *Domain Class* since they deal with real-world entities

Exercise

- Identify the *Entity* classes in Business Credit Disbursal

Entity Classes in Business Credit Disbursal...

- Fix appointment with customer
  - Telemarketing Executive, DME (User Information)
  - Credit card lists, loyality program lists, own address book (List Information)
  - Appointment (Appointment details)
  - Customer (Customer Information)

Entity Classes in Business Credit Disbursal...

- Meet customer and explain credit products
  - DME (User Information)
  - Appointment (Appointment details)
  - Customer (Customer Information)
  - Credit Products (Product Information)

Entity Classes in Business Credit Disbursal...

- Fill application form
  - Application Form (Application Details)
  - Customer (Customer Information)

Entity Classes in Business Credit Disbursal...

- Collect application and mandatory documents
  - Application Form (Application Details)
  - Document (Document Information)
  - Appointment (Appointment details)

Entity Classes in Business Credit Disbursal...

- File application with credit officer
  - Application Form (Application Details)
  - Document (Document Information)
  - Credit Officer (User Information)

Entity Classes in Business Credit Disbursal

- Upload Prospect List
  - Partner
  - Prospect List File

Entity Classes in Business Credit Disbursal...

- Additional Classes
  - Document Definition for a product
  - Documents collected for an application

Boundary Classes

- Handle the communication between the system surroundings and the inside of the system
- Provide interfaces to actors
- Examine each actor-use case to discover boundary classes
- Facilitate communication with other systems

Exercise

- Identify the *Boundary Classes* in Business Credit Disbursal

Boundary Classes in Business Credit Disbursal...

- Fix appointment with customer
  - Maintain user entitlement
  - Upload lists
  - Create/Confirm/Cancel appointments
  - Maintain Customer information

Boundary Classes in Business Credit Disbursal...

- Meet customer and explain credit products
  - Maintain user entitlement
  - Confirm / Change appointment
  - Record follow-ups with customer
  - Maintain various credit products

Boundary Classes in Business Credit Disbursal

- Fill application forms
  - Enter/Modify application for credit facility
  - Enter/Modify documents collected
- File application with credit officer
  - Maintain user entitlement

Control Classes

- Model sequencing behavior
- Co-ordinate the events needed to realise the behavior specified in the use case
- If a control class is doing more than sequencing, then it is doing too much!

Exercise

- Identify the *Control Classes* in Business Credit Disbursal

Control Classes in Business Credit Disbursal

- Application flow (menu)

Stereotypes & Documentation for classes

- Stereotypes : Entity, Boundary, Control
- Documentation should state the purpose of the class and not the structure
- If you find it difficult to document the class then your abstraction is not good enough

Packages

- Groups classes together
- Each Package contains an interface that is realised by a set of public classes
- Public classes – those classes to which classes in other packages talk
- Implementation classes – those that do not talk

Exercise

- Identify the *Packages* in Business Credit Disbursal

Packages in Business Credit Disbursal

- Application
  - Application Details
  - Application Documents
- Product
  - Product Information
  - Product Documents

Packages in Business Credit Disbursal

- Administration & Security
  - User Information
- DSA
  - Appointment details
  - Customer Information
  - Document Information
  - List Information
- Interfaces
  - All boundary classes

Class Diagrams

- Main Class diagram is the logical view of the model
- Each package can have its own “Main” class diagram which typically displays the public classes of the package

Exercise

- Create class diagrams for all packages

View Of Participating Classes

- Class diagrams in use case view
- Attached to a use case
- Depict all classes that collaborate to implement the use case

Exercise

- Create VOPC for all use cases
- Write the program specifications for any one of the use cases

DISCOVERING OBJECT INTERACTION

Use case realisation…

- The *use case* diagram presents the outside view of the system
- Functionality of the use case captured in the *flow of events*
- *Scenarios* are used to describe how use cases are realised as interactions among societies of objects
- A scenario is an instance of an use case – one path through the flow of events of the use case

Use case realisation…

- Scenarios developed to identify the objects, classes and object interactions needed to carry out a piece of functionality specified by the use case
- Scenarios document decisions about how the responsibilities specified in the use case are distributed among the object and classes
- Each use case is a collection of scenarios – primary (happy flow) and Secondary (alternate / whatif / exception flows)

Use case realisation

- Early stages of analysis identify only the *primary scenarios*
- When you discover that your scenarios are repeating a lot of steps from your earlier primary scenarios then you have reached the finish line!
- Identify and document 80% of primary scenarios along with a good representation of secondary scenarios in the elaboration phase

Documenting Scenarios

- Sequence diagrams
- Interaction diagrams

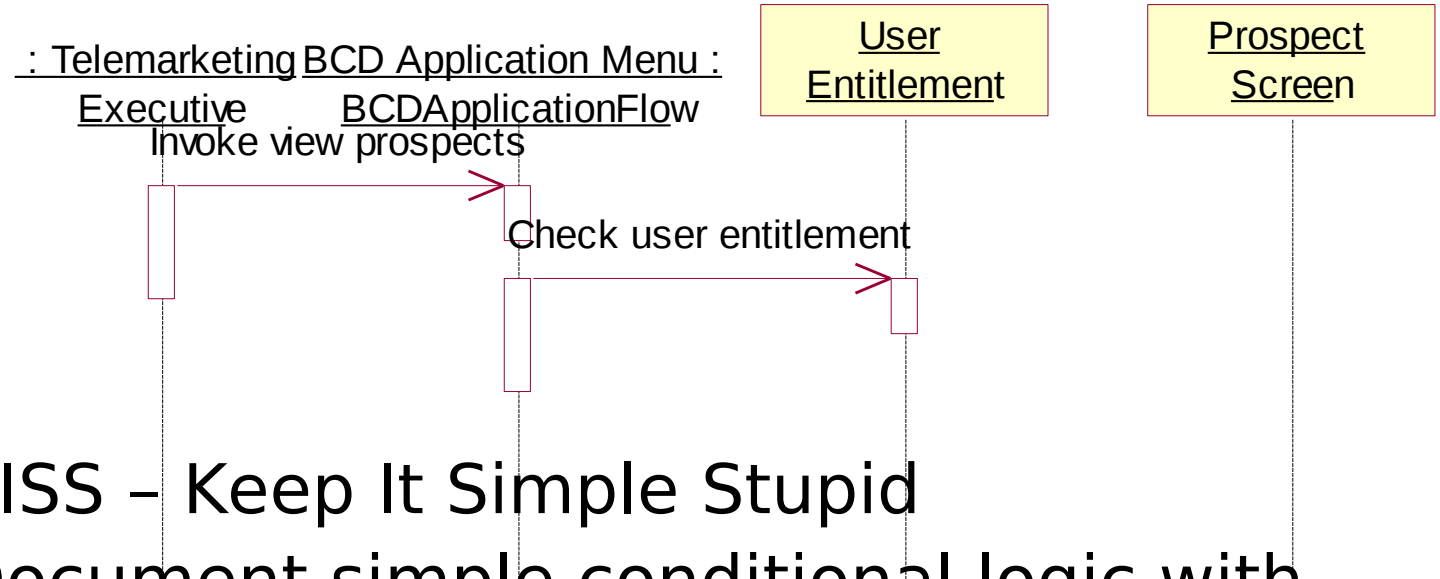
Sequence Diagrams

- Shows object interactions arranged in time sequence
- Depicts objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario

Exercise

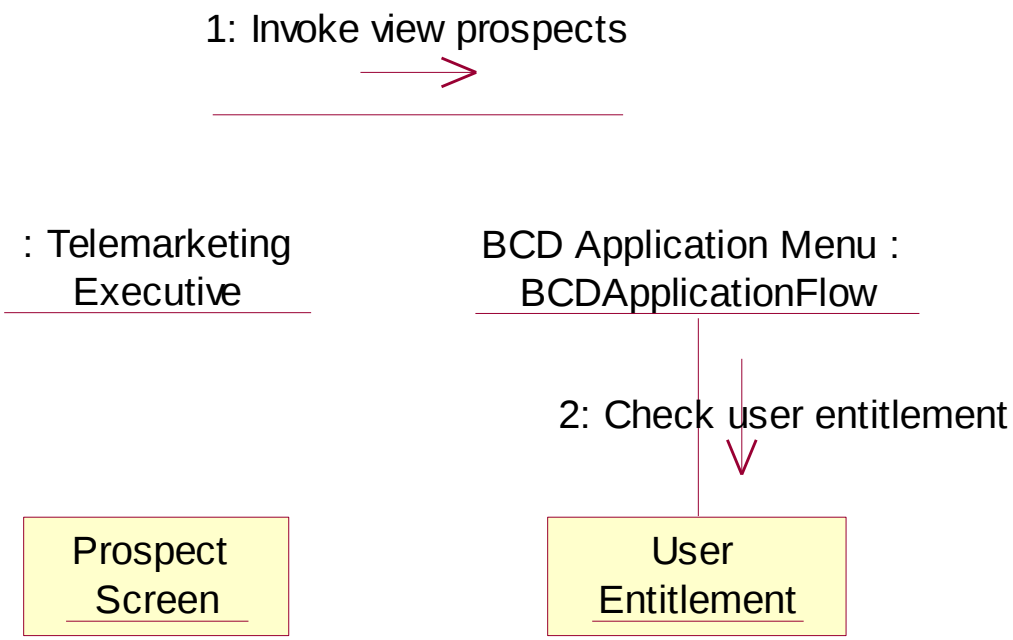
- Create sequence diagrams for the use cases
  - Drag the actors onto the sequence diagram
  - Click the object icon, click the area where you want the object to be placed
  - Drag the class onto the object

Sequence Diagram



- KISS – Keep It Simple Stupid
- Document simple conditional logic with notes and scripts otherwise use separate diagrams

Collaboration Diagram



- Alternate way to show a scenario
- Depict the big picture rather than in a time based order

Exercise

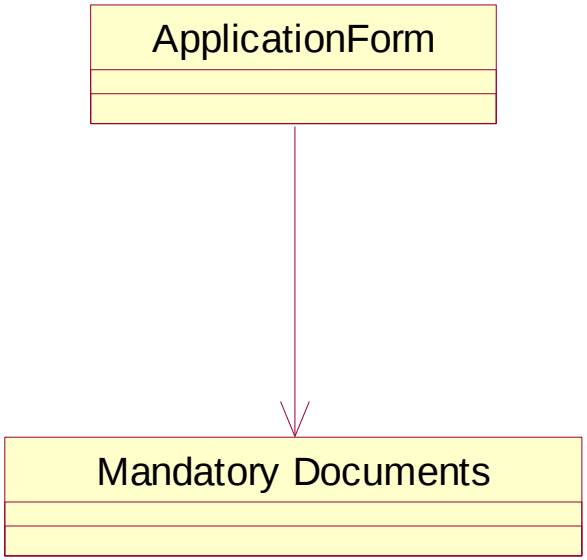
- Create the collaboration diagrams from the sequence diagrams

SPECIFYING RELATIONSHIPS

Need for relationships

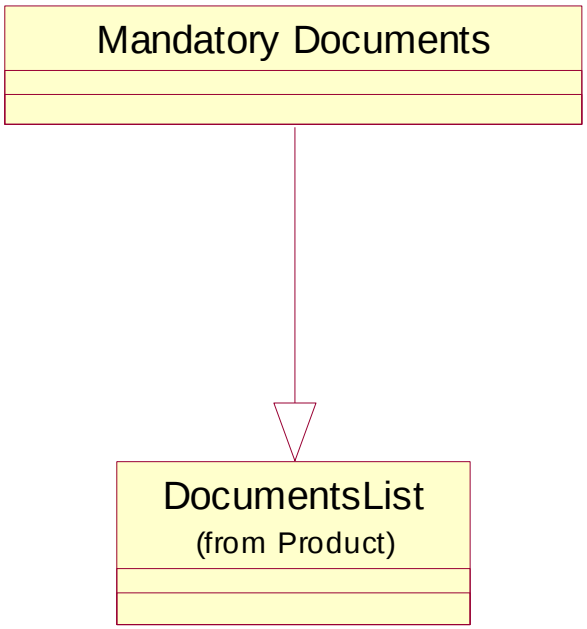
- System behavior is achieved through collaboration of objects and classes
- Relationships
  - Associations
  - Aggregations

Association



- Bidirectional semantic connection between classes
- Not “data flow” as defined in structured analysis and design

Aggregation

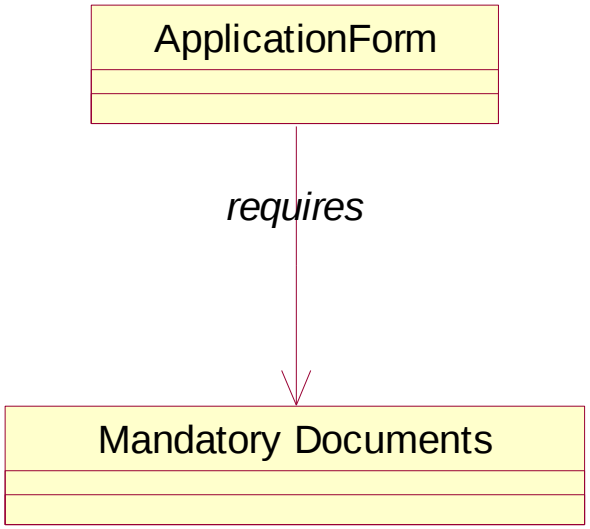


- Specialised form of association where a whole is related to its part(s)
- The arrow head should be at the “whole” end

Association or Aggregation?

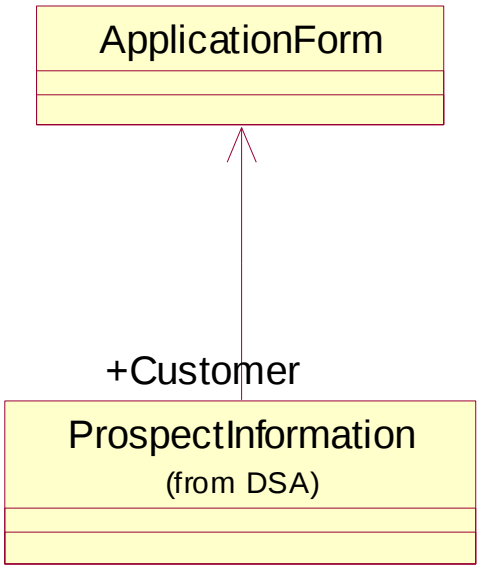
- Type of relationship between the bank and a loan
  - When you are modelling a bank ?
  - When you are modelling a loan application system ?

Naming Relationships



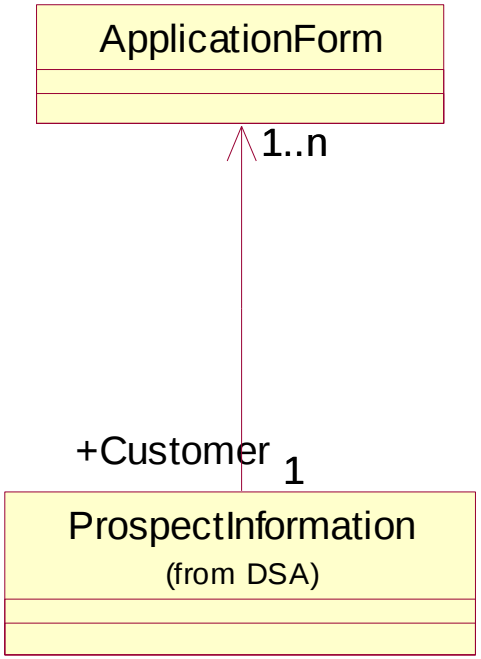
- Named with an active verb or verb phrase that communicates the meaning of the relationship
- The verb phrase implies the reading direction
- Name the association so it reads correctly from left to right or top to bottom
- Aggregation relationship or not usually named since it is read with the words “has” or “contains”

Role Names



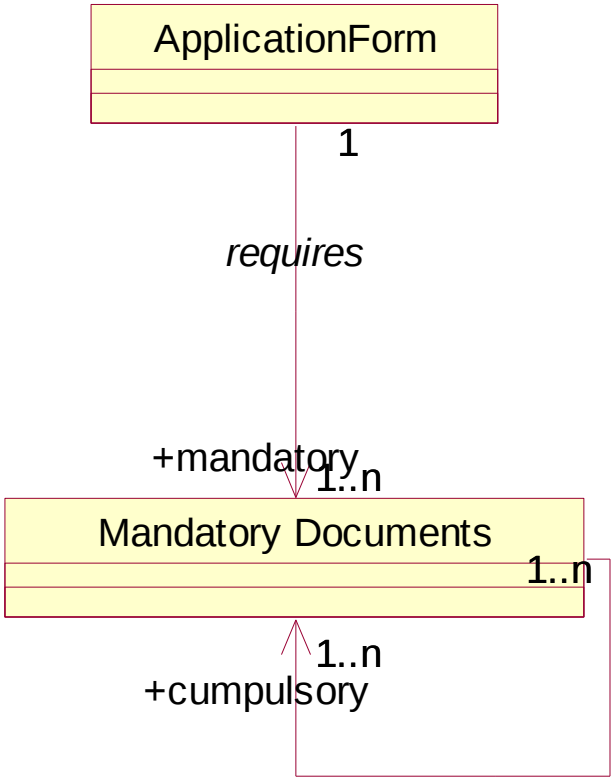
- The end of an association where it connects to a class is called an association role
- Role names can be used instead of association names
- A role name is a noun that denotes the purpose or capacity where one class associates with another
- Role name is placed on the association near the class it modifies
- It can be placed on one or both ends of an association line
- Not necessary to have both association and role name

Multiplicity Indicators



- Specified for classes
- Defines the number of objects that participate in a relationship
- Defines the number of objects that are linked to one another
- Multiplicity Indicators
  - 1
  - 0..*
  - 1..*
  - 0..1
  - 5..8
  - 4..7,9

Reflexive Relationships



- Multiple objects belonging to the same class may have to communicate with one another.
- This is shown as a reflexive association or aggregation
- Role names rather than association names typically are used for reflexive relationships

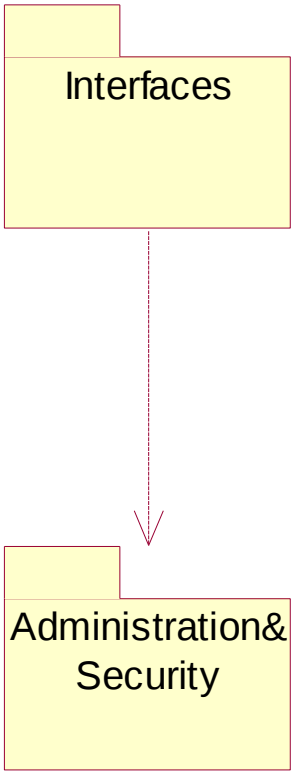
Finding Relationships

- Examine scenarios to determine relationships.
- Messages between objects mean that they need to communicate and hence need to have a relationship
- Associations and aggregations provide a pathway for communication
- Can also be determined based on the signature of an object (discussed later in this training)

Relationships in Business Credit Disbursal

Sending Class	Receiving Class	Association Type
BCD Menu	To all interface classes	Association
All interface classes	To entity classes	Association
Enter/Modify Application	Document Window	Association
Application Form	Mandatory Documents	Association
Mandatory Documents	Document List	Aggregation
List Providers	Prospect List	Association
Prospect List	Prospect	Association
Prospect	Appointment Details	Association
Product Information	Document List	Association

Package Relationships



- Is a dependency relationship
- If a one or more classes in a package initiates communication with one or more public classes in another package then there is a dependency
- Package relationships discovered by examining scenarios and class relationships

ADDING BEHAVIOUR AND STRUCTURE

Representing behavior and structure...

- A class embodies a set of responsibilities that define the behavior of the objects in the class
- The responsibilities are carried out by the operations defined for the class
- An operation should do only one thing, and it should do it well!

Representing behavior and structure

- The structure of an object is described by the attributes of the class
- Each attribute is a data definition held by objects of the class
- Objects defined for the class have a value for every attribute of the class
- Attribute values do not have to be unique
- Style guides (naming conventions) should be followed for defining attributes and operations – Avoid underscores
- If an object in a class does not need an attribute or operation, look at the class definition. The class is not cohesive and should be broken up into separate classes

Creating Operations...

- Messages to interaction diagrams are mapped to operations of the receiving class
- There are a few cases where a message does not become an operation
  - boundary classes where messages is a statement of the requirements for the GUI and usually represented as a GUI control (push button)
  - message between actors and interfaces (should be captured in user manual)
- Messaqes between actors that represent external systems create a class to hold the protocol that is used to perform the communication

Creating Operations

- Operations should be named in terms of the class performing the operation and not the class requesting the functionality
  - Create Application instead of Fill Application
- Operation names should not reflect the implementation of the operation
  - Appraisal by Credit Officer instead of Calculation of Creditworthiness
- Operations can be created independent of sequence diagrams
  - Incase there are operations that aid other operations e.g verify user entitlements

Relationship and Operation signatures…

- Signature of an operation may indicate a relationship.
- Fundamental Class for an *argument* or *return* from an operation (e.g. String) is not shown
- Other class relationships are displayed on Class diagrams

Relationship and Operation signatures

- Model these relationships as associations and then refine them into dependency relationships
- Also review the package relationships

Relationships based on operation signatures in BCD

- Application – Document, Prospect
- Appointment details – Prospect, User (DSA, DME)
- Documents – User (DME)
- Package Relationships – Application depends on DSA

Creating Attributes

- Many attributes found in the problem statement, requirements, flow of events documentation
- Also discovered when supplying the definition of a class
- Domain expertise and subject matter experts

Attributes in BCD…

- User Information – Id, Name, Password, +ve Financial Limit input, +ve Financial Limit authorisation, -ve Financial Limit input, -ve Financial Limit authorisation
- Application – Id, Name, Address, Amount

Attributes in BCD

- Mandatory Documents – Id, Name, Description, Purpose
- Appointment Details – Id, Date, Time, Prospect, Place, DME, Purpose
- Prospect – Id, Name, Address, Telephone, Email
- Prospect List – Partner id, List Name
- Document List – Id, Name, Description, Purpose
- Product Information – Id, Name, Min Amt, Max Amt, Doc Id

Displaying attributes and operations

- Create class diagrams and add the classes (Query/Add classes)
- Filter Relationships in Rose (Query/Filter)
- Display some attributes (Edit/Compartment)

Association Classes

- Relationship may have a structure and behavior
- E.g A customer can apply for any number of loans. Each Loan requires a set of documents. These documents don’t belong to the customer or the application. So contained in a class called document list
- Modeled as an association class

DISCOVERING INHERITENCE

Inheritance…

- One class shares the structure and behavior of one or more classes
- Also called *is-a* or *kind-of* hierarchy
- Attributes, operations and relationships are defined at the highest level of hierarchy at which they are applicable (so that descendent classes inherit them)
- Sub-classes can be augmented with additional attributes, operations

Inheritance…

- Sub class can supply its own implementation of an operation – Polymorphism
- Inheritance is not a relationship between different objects and hence not named. Role names are not used. Multiplicity does not apply.

Inheritance

- No limit to the depth of the hierarchy
- Limit it between 3-5 levels
- Inheritance key to reuse
- Finding Inheritance – Generalisation and Specialisation

Generalisation…

- Creating super classes that encapsulate structure and behavior common to several classes
- Common in beginning analysis endeavors since classes that currently exist model the real world
- Examine classes for commonality of structure and behavior

Generalisation

- Lookout for synonyms – attributes and operation names are expressed in natural english and the commonality might be hidden
- Look at attributes and behavior that at first glance are specific but in reality are general

Inheritance in BCD

- Telemarketing Executive and Direct Marketing Executive – USER
- Prospect and Customer

Specialisation

- Create sub classes that represent refinements to the super class
- This method comes to play if the classes already exist
- Sub classes should not restrict an operation defined in the super class I.e. should not provide less behavior

Inheritance in BCD

- Campaign details can be derived from a Product

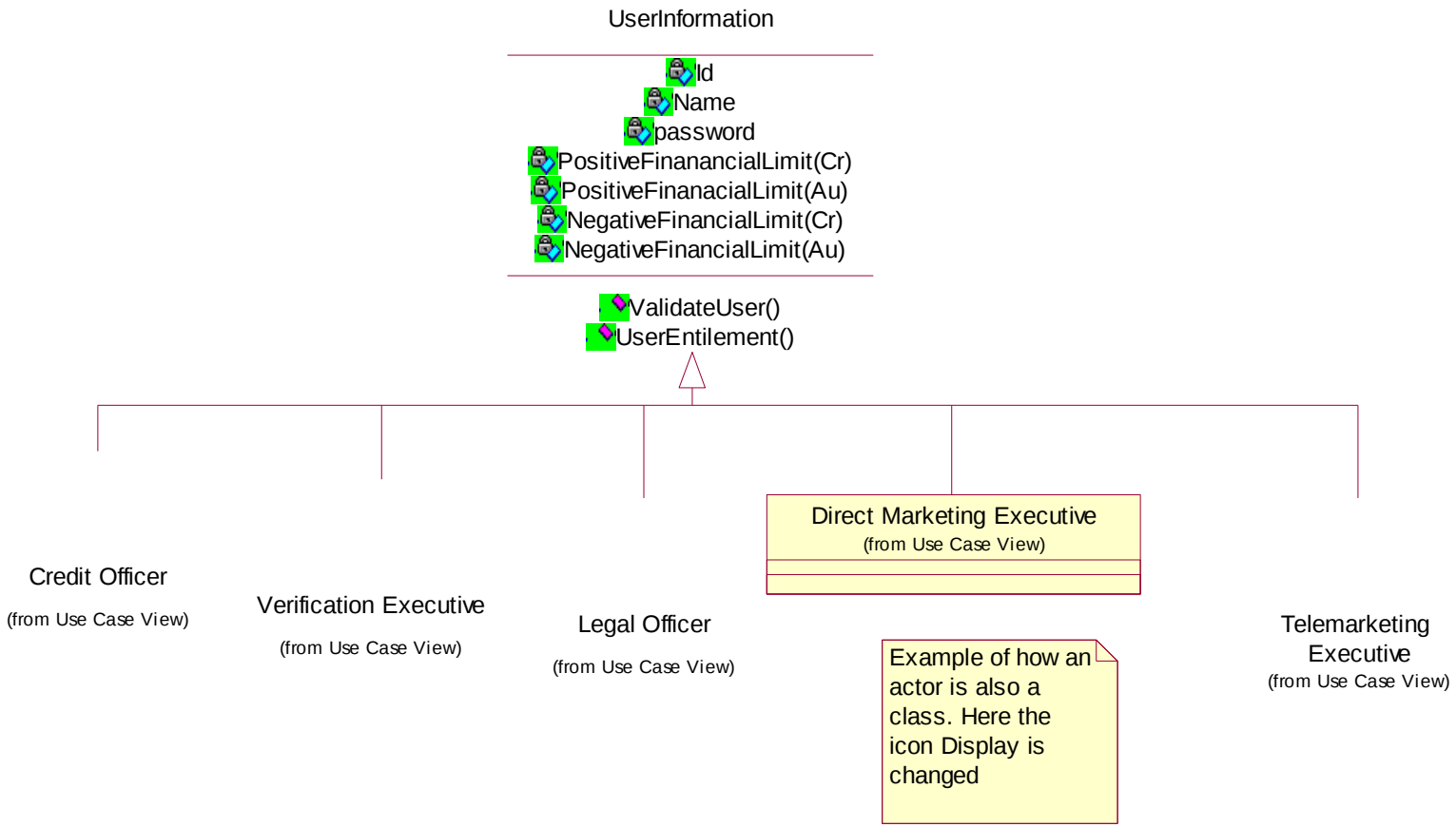
Inheritance Trees…

- The basis for specialisation is called *Discriminator*
- Discriminator has finite set of values and sub classes can be created for each value
- Inheritance relationship can be shown as a tree for all classes created from one discriminator

Inheritance Trees in BCD

- 1st User Discriminator : Financial Limit
- 2nd User Discriminator : Function (dept)
- Take care while discovering multiple discriminators
  - Do we have a multiple inheritance situation?
  - Should aggregation be used?
  - (As analysis and design progresses, the answers to these questions will lead to the structure of the model)

Inheritance Tree in BCD



Inheritance Trees

- Attributes, operations and relationships are relocated to the highest applicable level

Inheritance Tree in BCD…

- Attributes – User id, password, financial limits at super class I.e. user level
- All these classes – Credit officer, verification executive have relation ship with the application

Inheritance Tree in BCD…

- What do we do?
  - Keep the relationship at the subclass level?
  - Move the relationship to super class with a constraint stating that one user information object should be a credit officer another one must be a verification executive
  - Which is correct?

Single Inheritance Vs Multiple Inheritance

- Single Inheritance – One set of parents
- Multiple Inheritance – More than one set of parents
- Avoid unless it is very very very… necessary

Inheritance Vs Aggregation

- Inheritance is often misused
- Users
  - Data Entry Operators (to create)
  - Managers (to authorise)
- What happens if a manager needs to create a record?
- Inheritance – used to separate commonality from specifics
- Aggregation – used to show composite relationships

Analysing Object Behavior

Modeling Dynamic Behavior

- Use Cases and Scenarios
  - Describe system behavior, I.e. interaction between objects in the system
- State Transition Diagram
  - Shows the states of a single object, the events or messages that cause a transition from one state to another
  - and the actions that result from a change in that state

State Transition Diagram...

- Not created for every class in the system
- Created only for classes with significant dynamic behavior
- Useful to investigate the behavior of an aggregate whole class and control classes

State Transition Diagram

- Study interaction / Sequence diagrams to determine the dynamic objects – once receiving and sending many messages
- Stay in an analysis frame of mind – concentrate on the what of the problem and not how of the solution

State Transition diagrams in BCD

- Prospect Screen

States

- A condition during the life of an object when it satisfies some condition, performs some action, or waits for an event.
- Maybe characterised by
  - one or more attributes of the class
  - Existence of a link to another object

State Transition Diagram

- Encompasses all the messages that an object can send and receive
- Scenarios represent one path through a state transition diagram
- Interval between two messages sent by an object typically represents a state
- Look at the space between the message lines in a sequence diagram

State Transition

- Represents a change from an originating state to a successor state
- Two ways to transition
  - automatic (activities of originating state completes)
  - non automatic (caused by named events)
- Transition are considered to take 0 time and cannot be interrupted

Special States

- Start State (Only one)
- Stop State (multiple)

State Transition Details...

- State transition
  - may have an action or guard condition associated with it
  - May trigger an event
- Action / Activity: Behavior that occurs during a transition
- Event : Message sent to another object in the system
- Guard Condition : Boolean Expression of attribute values that allows a state transition only if condition is true

State Transition Details

- Actions and Guards
  - Behaviors of objects and hence become operations
  - These are private operations and used only by the object

State Details…

- Actions that accompany all state transitions into a state to be placed as an entry actions within the state
- Actions that accompany all state transitions out of a state to be placed as an exit actions within the state

State Details

- Activity : Behavior that occurs within a state
- Starts when state is entered and either completes or is interrupted by an outgoing state transition
- This behavior is mapped to operations of the object

Entry, Exit and Activities in BCD

- Create / Modify / Cancel Appointments